УДК 62; 004; 007 DOI: 10.14529/ctcr200402

РЕТРОСПЕКТИВА РАЗВИТИЯ ВЕБ-ТЕХНОЛОГИЙ В СОЗДАНИИ КОРПОРАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

А.А. Шинкарев
ООО «Софтмаст-ИТ», г. Челябинск, Россия

Введение. Первые упоминания о корпоративных информационных системах появились в 60-х гг. XX в. Со временем эти системы развивались, становились более сложными, но при этом позволяли решать широкий круг задач. Начиная с 90-х гг. XX в. стали активно развиваться веб-технологии. Естественным образом они нашли свое применение и при разработке корпоративных информационных систем различного назначения. **Цель исследования**. Описание основных этапов развития веб-технологий, начиная от появления языка гипертекстовой разметки до современных одностраничных веб-приложений, а также их влияния на подходы к созданию корпоративных информационных систем. Выявление перспективных направлений развития решений на базе веб-технологий, которые могут успешно использоваться при построении корпоративных информационных систем. **Материалы и методы**. Рассматриваются современные веб-технологии, анализируется процесс их развития от этапа появления до настоящего времени, рассматриваются примеры инструментов, которые были признаны устаревшими, и причины невозможности дальнейшего развития этих технологий. **Результаты**. В статье приводится история появления различных технологий, описывается их влияние на подходы к реализации корпоративных систем, а также приводится способ перехода от настольной версии системы к ее онлайн-версии. В том числе дается оценка перспективности отдельных технологических направлений, которые имеют хорошие шансы успешно развиваться в дальнейшем. Дается оценка растущей сложности веб-приложений, их сращивания с подходами и инструментами разработки серверной части. Выявляется актуальность разработчиков программного обеспечения, которые могут разрабатывать как серверную, так и клиентскую часть приложения.

Ключевые слова: веб-технологии, корпоративные информационные системы, браузер, веб-сервер, одностраничные приложения, JavaScript, ERP.

Введение

В течение последних десяти лет произошли значительные изменения в том, как мы создаем программное обеспечение. Этот эволюционный процесс затронул не только серверную часть с массовым переходом к использованию доступных облачных технологий и широким выбором технологий хранения данных, отличных от традиционных реляционных систем управления базами данных. Во многом изменился и подход к созданию пользовательских интерфейсов. Большой акцент сегодня сместился именно на веб-приложения. Они продолжают, как и прежде, использоваться для продвижения публичных интернет-ресурсов. Но что важнее, веб-приложения становятся одним из основных способов реализации решений куда более широкого спектра задач. Это и корпоративные сервисы, которые раньше в большей степени реализовывались через настольные приложения (Desktop Applications). Разработка мобильных приложений также во многом стремится использовать всю мощь современных браузеров. Даже те приложения, которые выглядят и устанавливаются как классические настольные приложения, на самом деле технически являются веб-приложениями, запущенными в браузере, лишенном классических элементов управления и навигации в сети Интернет. Примерами таких приложений сейчас являются Skype и Slack. Реализация приложений в виде веб-решений позволяет сделать их кроссплатформенными, то есть их можно запустить на любой из современных операционных систем, как и в основном в любом современном браузере.

1. Основные этапы развития веб-технологий

В начале 1990-х гг. Интернет существовал в виде протоколов, которые позволяли обмениваться сообщениями и письмами. Первый сайт (info.cern.ch) появился лишь в конце 1990 г. и со-

стоял полностью из текста. Так зародился HTML (HyperText Markup Language – язык гипертекстовой разметки), основы которого были описаны на этом сайте. HTML довольно быстро стал поддерживать 16 цветов и позволял вставлять изображения. Появление таких возможностей стало толчком к появлению первых браузеров – Mosaic (1993) и NetScape (1994) [1, 2].

В 1991 г. появилась первая спецификация протокола HTTP 0.9 (Hyper Text Transfer Protocol – протокол передачи гипертекста). С помощью этой версии протокола можно было выполнять только GET-запросы (простейшие запросы на чтение). При этом он не содержал никаких заголовков, поэтому с его помощью можно было передавать только HTML-страницы. В 1996 г. появилась версия HTTP 1.0, в которой была реализована поддержка заголовков и кода состояния, а также появилась возможность передавать разный формат документов. Все эти нововведения появились не одномоментно, а постепенно в период с 1991 по 1995 г. С тех пор HTTP развивался и продолжает совершенствоваться по сей день. Он стал быстрее, надежнее и универсальнее [3, 4].

В 1991 г. Тим Бернерс-Ли разработал первый веб-сервер CERN-httpd. По мере развития HTML, HTTP и Интернета в целом появилась потребность в более быстром сервере. Им стал NCSA httpd. Немного позднее к развитию этого сервера подключились другие разработчики, которые добавляли новую функциональность и улучшения с исправлениями. В 1995 г. все изменения были объединены в новый сервер, известный как Apache. По развитию веб-серверов можно судить о том, что Интернет и веб-технологии эволюционировали и становились мощнее и удобнее. Благодаря этому стали доступны те возможности Web, которыми мы обладаем сейчас [5, 6].

Марк Андрессен, основатель Netscape Communications и член бывшей команды Mosaic, считал, что Интернету нужен способ стать более динамичным. Анимация, взаимодействие и другие формы небольшой автоматизации должны стать частью сети будущего. Поскольку на тот момент веб-технологии находились на том уровне, чтобы быть понятными «непрограммистам», то и технология, которая бы сделала страницы Интернета более динамичными, должна была стать доступной обычным пользователям, а не только разработчикам. Так был создан JavaScript, который сначала назывался Mocha. Для разработчиков планировалось использование очень популярного в то время языка Java, а точнее Java-апплетов, которые могли выполняться в браузере. В 1994 г. появилась первая версия JavaScript. В течение десятилетия наиболее популярным его использованием был код такого вида (Листинг 1), что наглядно показывает, насколько ограниченным было изначальное предназначение языка JavaScript [7].

```
element.onClick = function() {
 document.getElementById("myImage").src = "image.jpg";
}
```

Листинг 1
Listing 1

Язык программирования JavaScript изначально не был разработан для создания сложного программного обеспечения. Это язык программирования с динамической типизацией, с синтаксисом, который прощает ошибки, неоднозначным механизмом наследования прототипа и отсутствующим понятием пакета или модуля. Все эти особенности первоначальных версий делают его более чем неоднозначным языком программирования, но именно ему посчастливилось оказаться в нужном браузере в нужное время для того, чтобы сегодня ему не было альтернатив. То, что начиналось как скриптовый язык для украшения HTML-страниц, сейчас используется обширнее, чем когда-либо. Несмотря на всю его распространенность, изначальные слабые стороны языка до сих пор позволяют писать код так, как этого делать не следует.

Помимо JavaScript в 1994 г. был создан язык программирования PHP в виде CGI-скриптов, написанный с помощью языка программирования C. Уже через год потребовалась большая функциональность, которая была реализована в PHP в 1995 г. Эта новая реализация была способна в том числе взаимодействовать с базами данных. Ее широкие возможности лежали в основе фреймворка, с помощью которого пользователи могли создавать простые динамические веб-приложения, такие как гостевые книги [8].

Для простых пользователей компания Microsoft выпустила HTML-редактор FrontPage. Среди непрофессионалов приложение получило огромную популярность: теперь любой человек в домашних условиях мог за несколько минут создать симпатичный сайт. При этом

FrontPage отлично взаимодействовал с Internet Explorer, что сделало его популярным в среде веб-разработчиков.

Веб-программы, напоминающие современные SPA-сайты (Single Page Application – одностраничные приложения), на самом деле существуют с середины 90-х. Это сайты, использующие Java-апплеты и Flash. Также интересным технологическим решением, появившимся в 2007 г., была технология Silverlight, позволявшая писать кроссплатформенный код на C#, запускаемый в браузере. Однако из-за запрета на запуск в браузере Safari данная технология уже в 2011 г. считалась отмирающей и после официального объявления о прекращении поддержки в 2015 г. стала активно выводиться из использования в программных продуктах [9].

Так называемые темные века веб-технологий продлились до 2005 г. В начале 2000-х гг. в браузерах стало использоваться малоизвестное API под названием XML HTTP Request (оно же XHR). Пример простейшего кода, позволяющий асинхронно загрузить и отобразить данные, приведен ниже (Листинг 2). Асинхронный JavaScript в сочетании с технологией XML появился в 2005 г.

```
var request = new XMLHTTPRequest();
request.onload = function() {
    alert(this.responseText);
};
request.open('get', 'endpoint.php');
request.send();
```

Листинг 2
Listing 2

Рассмотрим основные вехи развития экосистемы веб-разработки в XXI в.

1. JSON (2001) – появление формата представления данных JavaScript Object Notation, который сейчас является самым популярным форматом, используемым для разработки интернет-решений. Сам формат основан на объектно-литеральном синтаксисе языка JavaScript.

2. JsLint (2002) – появление инструмента контроля качества программного кода, написанного на языке JavaScript. Эта библиотека является предком до сих пор популярной библиотеки JSHint.

3. JsMin (2003) – появление инструмента минификации программного кода, написанного на языке JavaScript. Минификация может значительно сокращать время, которое необходимо для загрузки ресурсов веб-сайта за счет сокращения размера итогового JavaScript-файла, что в свою очередь позволяет сайтам загружаться быстрее. В том числе в рамках агрессивной минификации производится так называемая аглификация (uglification), становится значительно сложнее читать код и разбираться в нем.

4. jQuery (2006) – создание библиотеки, позволяющей писать кросс-браузерный код, расширяющей функциональность стандартной библиотеки языка JavaScript, а также добавляющей библиотеку базовых UI-компонентов.

5. JavaScript: Сильные стороны (2008) – издание книги, оказавшей значительное влияние на сообщество веб-разработчиков и на то, как они пишут программный код.

6. NodeJS (2009) – релиз платформы, позволяющей запускать программный код на языке JavaScript на сервере. С этого момента язык перестает существовать только в песочнице браузера и начинает распространяться и на бекенд-разработку и обрастать огромным количеством различных фреймворков и библиотек, которые на сегодняшний день покрывают почти любую необходимость при разработке корпоративных и публичных веб-решений.

2. Переход от статических веб-страниц к одностраничным приложениям

В начале 2000-х гг. и на более поздних этапах развития интернет-технологий произошло много событий. К тому времени веб-сайты нашли применение во многих сферах повседневной жизни. Информация приобрела большое значение. В основе большинства сайтов лежал статический HTML с использованием некоторых серверных технологий, история появления которых была описана ранее. С приходом в мир веб-технологий библиотеки jQuery в 2006 г. было положено начало развитию SPA-решений. Следует отметить, что в названии этой технологии используется слово «application», а не «website», что может свидетельствовать о том, что использование веба вышло на новый уровень.

Однако jQuery был ориентирован на пользовательский интерфейс и не подходил для обработки данных приложения. Следующим логичным шагом развития веб-приложений становится реализация библиотеки KnockoutJS, выпуск первой версии которой состоялся в 2010 г. Она реализовывала паттерн проектирования MVVM (Model-View-ViewModel) и позволяла связывать данные и их представление в пользовательском интерфейсе [10]. Тем не менее создание полноценного SPA еще представлялось маловозможным. Незадолго до KnockoutJS в 2009 г. была представлена библиотека Backbone.js, которая предлагала полноценную среду для разработки максимально приближенного к SPA приложения. В 2010 г. появилась первая версия AngularJS – фреймворка, объединившего лучшие подходы в создании SPA и позволившего реализовывать их в том виде, в котором мы их знаем сейчас. AngularJS реализовывал двустороннюю привязку данных, клиентский MVC, шаблоны и внедрение зависимостей в одном фреймворке [11].

В современной разработке клиентской части приложений широкое распространение получило применение упомянутого ранее паттерна проектирования MVVM, развивающего паттерн разработки MVC (Model View Conroller) и возможно лучше подходящего для разработки пользовательских интерфейсов. В современных SPA-фреймворках присутствует двустороннее или одностороннее связывание данных (Data Binding). Для организации асинхронного взаимодействия используются промисы (Promises) или же реактивное программирование (Reactive Programming). Также инъекция зависимостей (Dependency Injection) является неотъемлемой частью архитектуры современных веб-решений.

На сегодняшний день кажется вполне вероятным дальнейшее усиление роли SPA-приложений на рынке разработки прикладного программного обеспечения. Этот инструмент успешно зарекомендовал себя как для создания публичных сайтов с высокими требованиями производительности и поддержки кодовой базы, так и для построения корпоративных веб-приложений с богатой доменной логикой, являющейся частью клиентского приложения, так называемых толстых клиентов. Помимо публичных и корпоративных систем SPA-решения находят свое применение и в создании мобильных и десктопных приложений, потому что позволяют писать логику на популярных у разработчиков кроссплатформенных технологиях, таких как HTML, CSS и JavaScript [12].

Сравнительно недавно появилась концепция под названием Micro Frontends, которая нашла в последнее время более серьезную поддержку на уровне официальных стандартов таких технологий, как Web Components [13]. Суть идеи заключается в том, чтобы дать возможность запускать в браузере для одного и того же приложения несколько разных фреймворков работы с деревом DOM (Document Object Model) через унифицированный интерфейс взаимодействия. Поддержка этого варианта взаимодействия уже присутствует во всех ведущих SPA-фреймворках. Эта поддержка будет только расширяться со временем. Идея разделить монолитную клиентскую логику веб-приложения по сути своей заимствует идеи микросервисной архитектуры, только применительно к клиентской стороне программного обеспечения. Естественно, идея разделения приложения на изолированные части, общающиеся между собой через унифицированный протокол, добавляет сложности процессам разработки и поддержки веб-приложений. Однако вариант полной изоляции может подойти крупным проектам, в разработку которых вовлечено несколько отдельных команд разработчиков. Этот шаг позволит сделать рост сложности программного обеспечения, который происходит с ростом объема функциональности, более линейным, а не экспоненциальным.

Стоит отметить, что подход с использованием технологии iframe [14], существующий по меркам веб-разработки очень давно, до сих пор может успешно использоваться для разбиения приложения с общим интерфейсом на части, обладающие изолированностью и независимостью развертывания. Кажется, что на данном этапе подход со встраиванием через iframe более предпочтителен по сравнению с вариантом использования не до конца зрелой технологии Micro Frontends [15].

Стоит отметить, что и направление внедрения кода, написанного не на JavaScript, имеет свое продолжение, в частности через технологию WebAssembly [16]. Например, ее использует Blazor, являющийся частью платформы .NET. Это также является интересным вариантом дальнейшего развития клиентских веб-приложений со сложной логикой, вынесенной за пределы серверной части.

Помимо технологий интеграции крупных модулей, которые были рассмотрены выше, важную роль играют системы реализации модульности на уровне кода отдельно взятого приложения. Для разбиения кода на модули и их встраивания в код использовались такие пакеты, как AMD, RequireJS, CommonJS. Распространение пакетов происходило с использованием таких сервисов, как NPM и Bower. Основными средствами автоматизации являлись Grunt и Gulp. Создание сложных программных продуктов, таких как, например, офисные пакеты работы с документами, уже во многом переходит в веб-формат. В частности, рабочая версия статьи набиралась в онлайн-версии Microsoft Word.

Во многом можно сказать, что разработка пользовательского интерфейса веб-решений становится все больше похожа на разработку серверной части за счет активного внедрения статической типизации с использованием таких библиотек, как TypeScript и Flow. Распространена тенденция, когда разработчики фронтенд-решений переходят к разработке бекенд-сервисов на платформе NodeJS, ведь вся экосистема им знакома. Также и бекенд-разработчики переходят к работе над логикой уровня представления, встречая все больше знакомых концепций, которые были у них позаимствованы. В целом кажется, что тенденция на смешение ответственности и переход к универсальному типу разработчика (Full-Stack Developer) только будет набирать обороты в дальнейшем.

3. Использование веб-технологий в построении корпоративных информационных систем

Развитие информационных технологий неизбежно привело к информатизации большей части процессов, протекающих в обществе и бизнесе. Все развивающиеся компании рано или поздно столкнулись с необходимостью систематизации и структурирования информации, а также с автоматизацией процессов обработки этой информации. На начальном уровне было возможно использование различных настольных приложений для работы с информацией, однако с увеличением ее объемов стало ясно, что необходимо создание корпоративных информационных систем. Корпоративная Информационная Система (КИС) – это масштабируемая система, предназначенная для комплексной автоматизации всех видов хозяйственной деятельности компаний, а также корпораций, требующих единого управления [17].

Рассмотрим этапы развития корпоративных информационных систем.

1. 1960-е гг.: появление MRP (Material Requirements Planning) – планирование потребности в материалах.

2. 1980-е гг.: появление MRP 2 (Manufactory Resource Planning) – планирование производственных ресурсов.

3. 1990-е гг.: появление ERP (Enterprise Resource Planning) – набор интегрированных приложений, позволяющих создать интегрированную информационную среду для автоматизации планирования, учета, контроля и анализа всех основных бизнес-операций предприятия.

4. 2000-е гг.: появление CSRP (Customer Synchronized Resources Planning) – планирование ресурсов, синхронизированное с покупателем [18].

Использование веб-технологий в построении корпоративных систем началось на этапе появления ERP. Поскольку информационные технологии активно развивались, то компании, занимающиеся разработкой КИС, понимали, что удобство использования систем будет намного выше, если взаимодействие с этими системами будет доступно по сети Интернет. Изначально это взаимодействие ограничивалось использованием HTML и XML. Архитектура таких систем включала в себя наличие сервера HTML-интерфейса, сервера XML-данных, сервера приложения, а также сервера СУБД (система управления базами данных). Сервера HTML и XML реализовывались на основе Apache или Netscape [19].

По мере развития HTTP, HTML, средств для создания пользовательских интерфейсов, способов хранения и обработки данных, а также увеличения объема этих данных зародилась идея реализации распределенных информационных систем. Первым шагом на пути к этому стало использование ASP (Application Service Provider). Технология ASP позволяла обеспечивать доступ к информационной системе, которая была установлена на стороннем сервере. Такой подход зародился по причине того, что КИС уже стоили больших денег, что делало невозможным их покупку предприятиями малого и среднего бизнеса. Кроме того, значительно выросла и сложность самих систем, что приводило к огромным расходам на распространение программного обеспечения

среди конечных пользователей. Доступ с использованием ASP предоставляла компания SAP к своему продукту SAP R/3 [20].

Традиционная структурная схема локальной информационной системы выглядит следующим образом.

1. Интерфейс пользователя.
2. Ядро системы.
3. Информационный массив.
4. Интерфейс администратора.
5. Утилиты администратора.

Переход от локальной информационной системы (ИС) к ее веб-версии возможен при реализации ее структурных элементов с использованием веб-технологий. С точки зрения веб-технологии интерфейс пользователя – это браузер, который взаимодействует с ядром через HTTP-сервер. Таким образом происходит первый этап декомпозиции традиционной ИС в веб.

Второй шаг – это возможность использования браузера в качестве интерфейса администратора. Здесь возникают вопросы разграничения доступа и актуализации информации в базах данных системы.

Следующий шаг – распределение нагрузки по нескольким серверам, а также использование кэширования на серверах-посредниках.

Пока декомпозиции подвергалась связка «конечный пользователь – ядро». Можно провести декомпозицию и на стороне сервера. Первым таким шагом является применение CGI при доступе к ресурсам. Сервер становится посредником между браузером и сервером ресурса. Более эффективно это решается за счет API (Application Programming Interface), когда сам HTTP-сервер имеет модуль доступа к серверному процессу.

Другим важным моментом является внедрение результатов обращения к внешнему ресурсу в готовые шаблоны страниц. В терминологии веб – это Server Site Include. Вставка может осуществляться как локально, так и с использованием данных удаленного сервера. Таким образом, сервер оснащается языком манипулирования данными при формировании отклика.

Другой инструмент – это видоизменение страниц на стороне клиента. Управление формой интерфейса пользователя осуществляется на стороне клиента при помощи скриптовых языков, таких как JavaScript. В первую очередь, это позволяет повысить интерактивность ИС, построенной на основе веб-технологий. Сейчас активно используется SPA-подход в организации пользовательского интерфейса.

Для поддержки сеанса работы с сервером в веб применяется спецификация Cookie. Идея состоит в том, чтобы передавать от клиента на сервер и обратно информацию о пользователе и его действиях, которая привязывается по типу информационного ресурса и времени [21]. Например, компания SAP предоставляет ERP, реализованную по схожему подходу.

Таким образом, использование веб-технологий при разработке КИС позволяет в любое время и любом месте:

1) посылать письменные сообщения;

2) представлять свое предприятие, свои товары и услуги, осуществлять обратную связь с клиентами (получать заказы, рекламации и т. д.);

3) заниматься маркетингом;

4) искать сотрудников;

5) устанавливать деловые связи, покупать товары и услуги;

6) производить прямые расчеты с покупателями и поставщиками.

Переход ИС из громоздких и сложных локальных приложений на сторону веб-технологий позволяет предприятиям малого и среднего бизнеса в полной мере использовать возможности, которые сегодня предоставляют информационные технологии, а значит, выйти на новый уровень эффективности ведения бизнеса.

Заключение

Развитие веб-технологий сыграло важную роль в подходах к созданию корпоративных информационных систем. Технологии создания веб-приложений прошли большой путь развития от простейших текстовых страниц до больших и сложных приложений, способных решать широкий круг бизнес-задач.

По мере развития корпоративных информационных систем от простых настольных приложений до крупных системных комплексов со сложной архитектурой их стоимость на рынке программного обеспечения также росла, росли затраты на распространение систем между конечными пользователями в организации. По этой причине предприятия малого и среднего бизнеса не могли позволить себе использование корпоративных информационных систем.

Поскольку в современном мире сложилась тенденция перехода различных систем в веб-пространство, а также тот факт, что веб-технологии достигли высокого уровня развития, закономерно, что стала возможна и реализация корпоративных информационных систем с использованием новейших веб-технологий. Это в свою очередь открывает новые пути развития таких систем и дает возможность их использовать малым и средним компаниям для достижения бизнес-целей. Доступность системы, требующей от пользователя лишь наличия браузера, 24 часа 7 дней в неделю в публичном пространстве, в значительной степени определила лидирующую роль веб-решений по сравнению с классическими настольными приложениями.

Решения, базирующиеся на SPA-фреймворках, имеют все шансы продолжить господствовать среди инструментов создания веб-приложений. Однако не стоит сбрасывать со счетов концепцию микрофронтендов, которая, возможно, нивелирует синтаксическую пропасть между лидирующими SPA-фреймворками сегодня. Также возможным направлением универсализации является развитие таких технологий, как рассмотренная технология WebAssembly, запуска в браузере кода, написанного не на языке программирования JavaScript.

Бесспорно одно – разработка клиентской части веб-приложений будет продолжать усложняться и экосистема технологий вокруг будет лишь расти, вместе с тем рождая новые альтернативы реализации той же функциональности, но используя разные языки, библиотеки и подходы к интеграции. Сложность этой части программного обеспечения во многом будет сравниваться с реализацией серверной части, а актуальность программистов-универсалов на рынке труда будет лишь усиливаться.

Литература/References

1. Wallace J. *HTML5 Quick Markup Reference*. 1st ed. Edition. New York, Apressm, 2016. 257 p.

2. Daubs M. *The SAGE International Encyclopedia of Mass Media and Society*. Los Angeles, SAGE Publications, 2019. 2168 p.

3. Grigorik I. *High Performance Browser Networking: What every web developer should know about networking and web performance*. Sebastopol, O'Reilly Media, 2013. 400 p.

4. Evolution of HTTP. Available at: https://developer.mozilla.org/ru/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP (accessed 23.09.2020).

5. Berners-Lee T., Luotonen A. *CERN httpd Reference Manual*. Geneva, CERN, 1994.

6. Laurie B., Laurie P. *Apache: The Definitive Guide*. Sebastopol, O'Reilly Media, 2002. 536 p.

7. Peyrott S. It All Began in the 90s. *A Brief History of JavaScript*, 2017, pp. 2–6.

8. Lerdorf R., Tatroe K., MacIntyre P. *Programming PHP*. Sebastopol, O'Reilly Media, 2013. 528 p.

9. *Microsoft Silverlight*. Available at: https://en.wikipedia.org/wiki/Microsoft_Silverlight (accessed: 12.09.2020).

10. *Knockout (web framework)*. Available at: https://en.wikipedia.org/wiki/Knockout_(web_framework) (accessed 14.09.2020).

11. *AngularJS*. Available at: https://en.wikipedia.org/wiki/AngularJS (accessed 20.09.2020).

12. Mikowski M. *Single Page Web Applications: JavaScript end-to-end*. Shelter Island, Manning Publications, 2014. 432 p.

13. Web Components. Available at: https://developer.mozilla.org/en-US/docs/Web/Web_Components (accessed 20.09.2020).

14. <iframe>: The Inline Frame element. Available at: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe (accessed: 22.09.2020).

15. Geers M. *Micro Frontends in Action Paperback.* Shelter Island, Manning Publications, 2020. 296 p.

16. WebAssembly. Available at: https://en.wikipedia.org/wiki/WebAssembly (accessed: 24.09.2020).

17. Сатунина А.Е., Сысоева Л.А. Управление проектом корпоративной информационной системы предприятия. М.: Финансы и статистика: Инфра-М, 2009. 352 с. [Satunina A. E., Sysoeva L.A. *Upravleniye proyektom korporativnoy informatsionnoy sistemy predpriyatiya* [Project management of corporate information systems of the enterprise]. Moscow, Finansy i statistika, Infra-M, 2009. 352 p. (in Russ.)]

18. Тамьяров А.В. История развития и современные проблемы корпоративных информационных систем. Вестник Волжского университета им. В.Н. Татищева. 2011. № 30. [Tam'yarov A.V. [History of development and modern problems of corporate information systems]. Vestnik of Volzhsky University after V.N. Tatischev, 2011, no. 30. (in Russ.)]

19. Majchrzak A., Traverso P., Monfort V. *Web Information Systems and Technologies: 14th International Conference*. New York, Springer, 2017. 274 p.

20. Kelly L., Thomas J. *Application Service Provider and Software as a Service Agreements Line by Line: A Detailed Look at ASP and Saas Agreements and How to Change Them to Meet Your Needs*. Eagan, Aspatore Books, 2009. 108 p.

21. Vidgen R. *Developing Web Information Systems: From Strategy to Implementation*. Oxford, Butterworth-Heinemann, 2002. 274 p.

Шинкарев Александр Андреевич, канд. техн. наук, инженер-программист, ООО «Софтмаст-ИТ», sania.kill@mail.ru.

Поступила в редакцию 9 октября 2020 г.

DOI: 10.14529/ctcr200402

RETROSPECTIVE OF WEB TECHNOLOGIES EVOLUTION IN DEVELOPMENT OF ENTERPRISE INFORMATION SYSTEMS

***A.A. Shinkarev**, sania.kill@mail.ru*
Softmast-IT LLC, Chelyabinsk, Russian Federation

Introduction. The first mentions of enterprise information systems refer to the 1960s. These systems developed over time, becoming more complex and allowing to solve a wide range of problems. The 1990s mark the beginning of the active development of web technologies. Naturally, they found way into the development of enterprise information systems used for various purposes. **The purpose of the study** was to describe the main stages in the development of web technologies, from the appearance of the hypertext markup language to modern single-page web applications, as well as the impact they had on approaches to the development of enterprise information systems. The author meant to identify promising trends in web-based solutions that can be successfully used in the development of enterprise information systems. **Materials and methods.** The paper discusses modern web technologies, their development from the dawn to the present day, as well as some tools recognized as obsolete, and the reasons why it is impossible to develop them further. **Results.** The paper gives the history of the emergence of various technologies, describes their impact on approaches to the implementation of enterprise systems, and a way to move from a desktop version of the system to an online one. Some predictions are made concerning the prospects of certain technological areas that have good chances to successfully develop in the future. The author assesses the growing complexity of web applications, the way they merge with the server side development approaches and tools. The demand of software engineers who can develop both the server and client side of an application is rationalized.

Keywords: web technologies, enterprise information systems, browser, web server, single page application, JavaScript, ERP.

Received 9 October 2020

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Шинкарев, А.А. Ретроспектива развития веб-технологий в создании корпоративных информационных систем / А.А. Шинкарев // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника». – 2020. – Т. 20, № 4. – С. 14–21. DOI: 10.14529/ctcr200402

FOR CITATION

Shinkarev A.A. Retrospective of Web Technologies Evolution in Development of Enterprise Information Systems. *Bulletin of the South Ural State University. Ser. Computer Technologies, Automatic Control, Radio Electronics*, 2020, vol. 20, no. 4, pp. 14–21. (in Russ.) DOI: 10.14529/ctcr200402